SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 6 Data 07.12.2024 Temat: „Analiza danych z wykorzystaniem narzędzi do modelowania regresji” Wariant 2	Anna Więzik Informatyka II stopień, niestacjonarne, 1 semestr, gr.1b

1. Polecenie:

Link do repozytorium: https://github.com/AnaShiro/NoD2_2025

2. Wine: `https://archive.ics.uci.edu/ml/datasets/wine`

Analiza skupień to narzędzie eksploracji danych wykorzystywane do segregowania obiektów w grupy według stopnia ich podobieństwa. Dąży się do tego, by elementy w obrębie jednej grupy wykazywały większe podobieństwo między sobą niż z elementami z innych grup. W tym celu stosuje się różne miary odległości, takie jak: metryka euklidesowa, Manhattan czy cosinusowa.

W praktyce wykorzystywane są rozmaite techniki grupowania. Metoda K-średnich (K-means), będąca jedną z najczęściej stosowanych, redukuje wewnętrzną zmienność grup, choć wymaga wcześniejszego określenia liczby klastrów. Z kolei DBSCAN bazuje na analizie gęstości i świetnie sprawdza się w wykrywaniu struktur o nieregularnych kształtach i w obecności szumu.

Aby ocenić jakość wyników i dobrać odpowiednią liczbę skupień, stosuje się takie narzędzia jak: metoda łokcia, wskaźnik Silhouette, czy kryteria Calinskiego-Harabasza i Daviesa-Bouldina. Umożliwiają one identyfikację najbardziej trafnego podziału danych oraz porównanie efektywności poszczególnych metod grupowania.

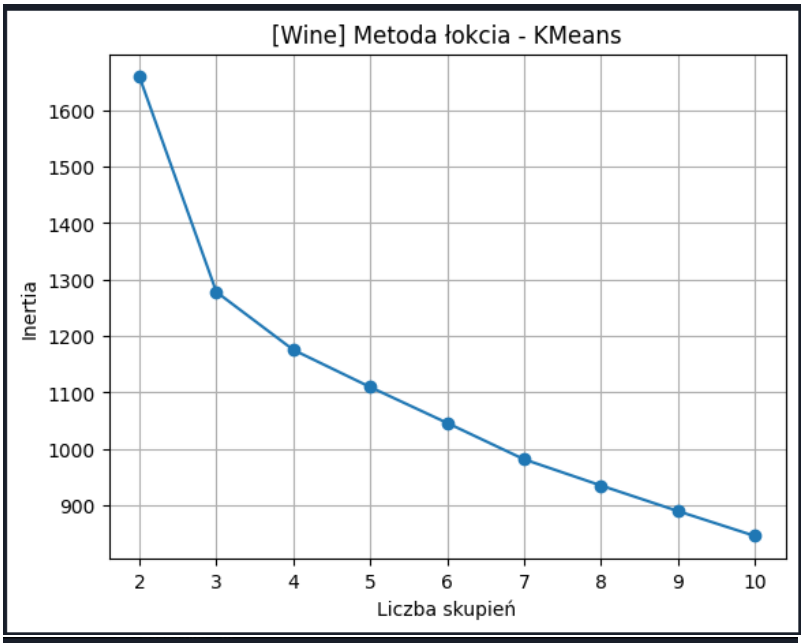
2. Opis programu opracowanego

```python
from sklearn.datasets import load_wine

data = load_wine().data
dataset_label = "Wine"
```

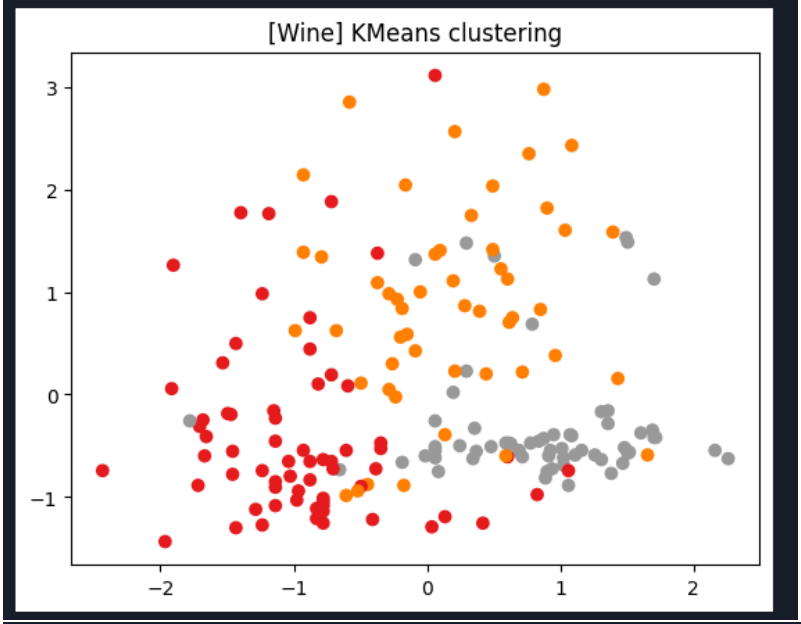
✓ 0.4s

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import matplotlib.pyplot as plt
import numpy as np

X = StandardScaler().fit_transform(data)

inertia = []
k_range = range(2, 11)
for k in k_range:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X)
    inertia.append(km.inertia_)

plt.plot(k_range, inertia, marker='o')
plt.title(f"[{dataset_label}] Metoda łokcia - KMeans")
plt.xlabel("Liczba skupień")
plt.ylabel("Inertia")
plt.grid(True)
plt.show()

kmeans = KMeans(n_clusters=3, random_state=42)
labels_km = kmeans.fit_predict(X)

sil_km = silhouette_score(X, labels_km)
ch_km = calinski_harabasz_score(X, labels_km)
db_km = davies_bouldin_score(X, labels_km)

print(f"[{dataset_label}] KMeans:")
print(f" - Silhouette Score: {sil_km:.2f}")
print(f" - Calinski-Harabasz: {ch_km:.2f}")
print(f" - Davies-Bouldin: {db_km:.2f}")

plt.scatter(X[:, 0], X[:, 1], c=labels_km, cmap='Set1')
plt.title(f"[{dataset_label}] KMeans clustering")
plt.show()
```

✓ 1.6s



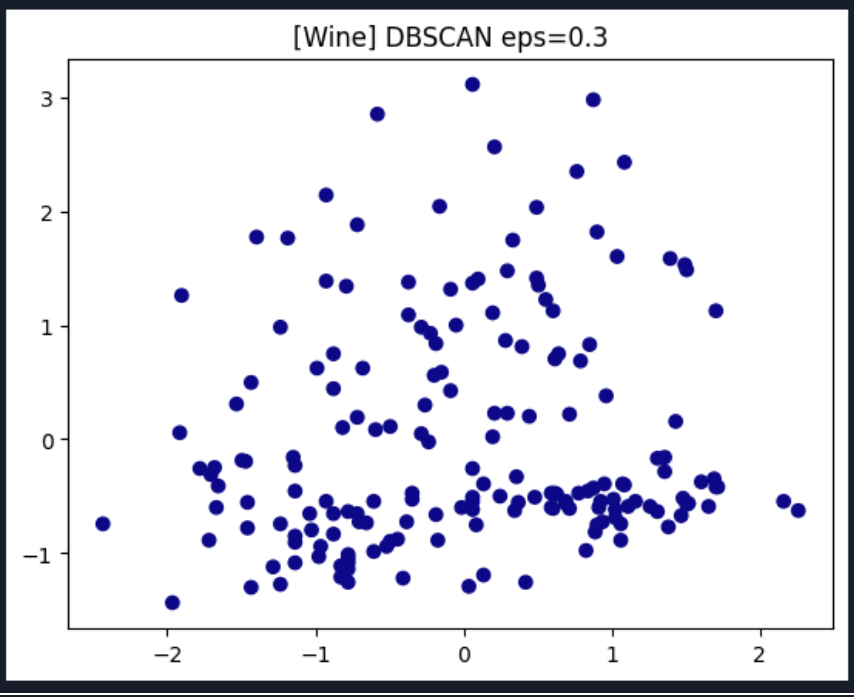

```
[Wine] KMeans:
 - Silhouette Score: 0.28
 - Calinski-Harabasz: 70.94
 - Davies-Bouldin: 1.39
```



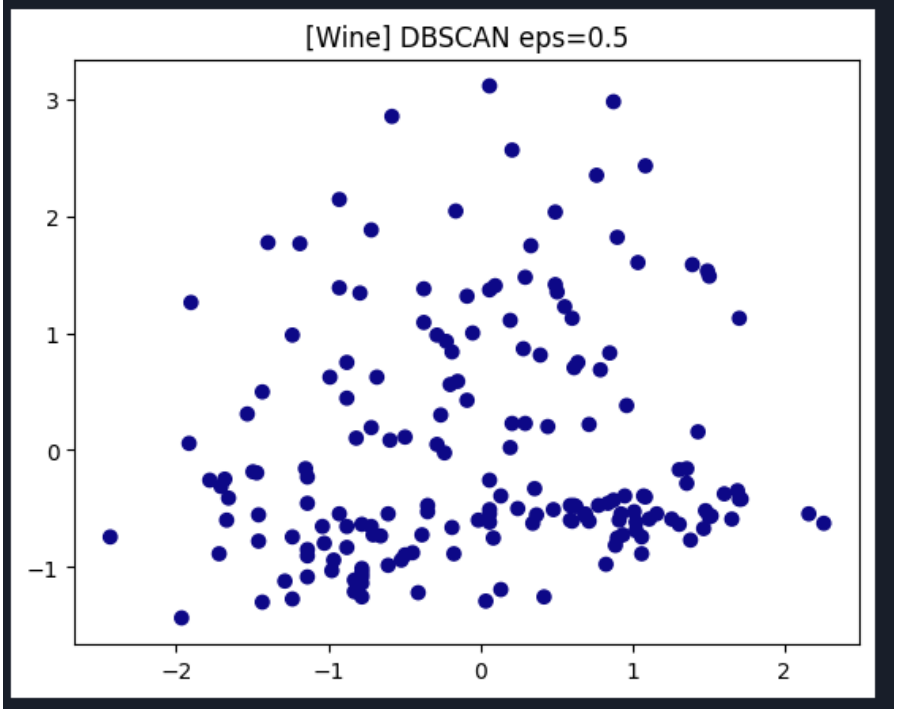

```python
from sklearn.cluster import DBSCAN

print(f"[{dataset_label}] DBSCAN:")
for eps in [0.3, 0.5, 0.7]:
    db = DBSCAN(eps=eps, min_samples=5)
    labels_db = db.fit_predict(X)
    n_clusters = len(set(labels_db)) - (1 if -1 in labels_db else 0)
    if n_clusters >= 2:
        sil_db = silhouette_score(X, labels_db)
        print(f" - eps={eps} → clusters={n_clusters}, silhouette={sil_db:.2f}")
    else:
        print(f" - eps={eps} → za mało skupień ({n_clusters})")

    plt.scatter(X[:, 0], X[:, 1], c=labels_db, cmap='plasma')
    plt.title(f"[{dataset_label}] DBSCAN eps={eps}")
    plt.show()
```

✓ 0.1s

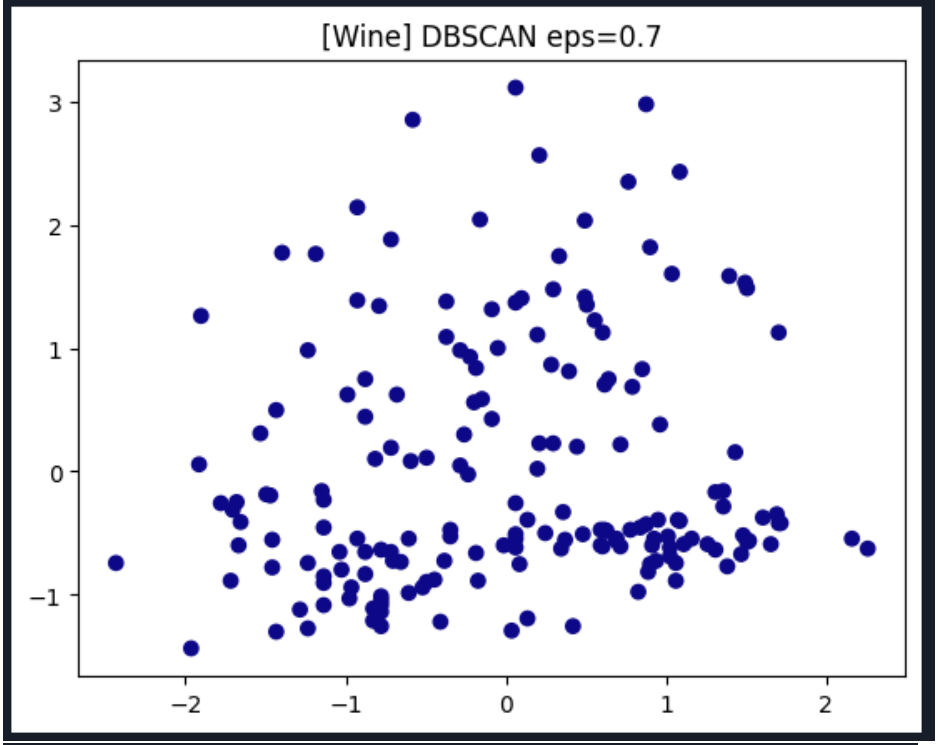
```
[Wine] DBSCAN:
 - eps=0.3 → za mało skupień (0)
```




```
- eps=0.5 → za mało skupień (0)
```

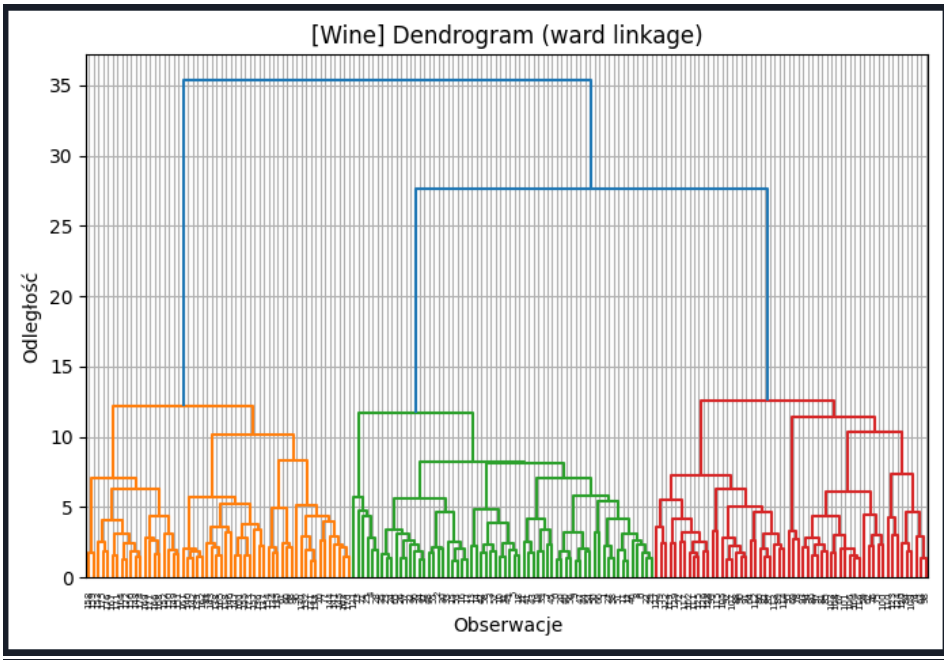


- eps=0.7 → za mało skupień (0)

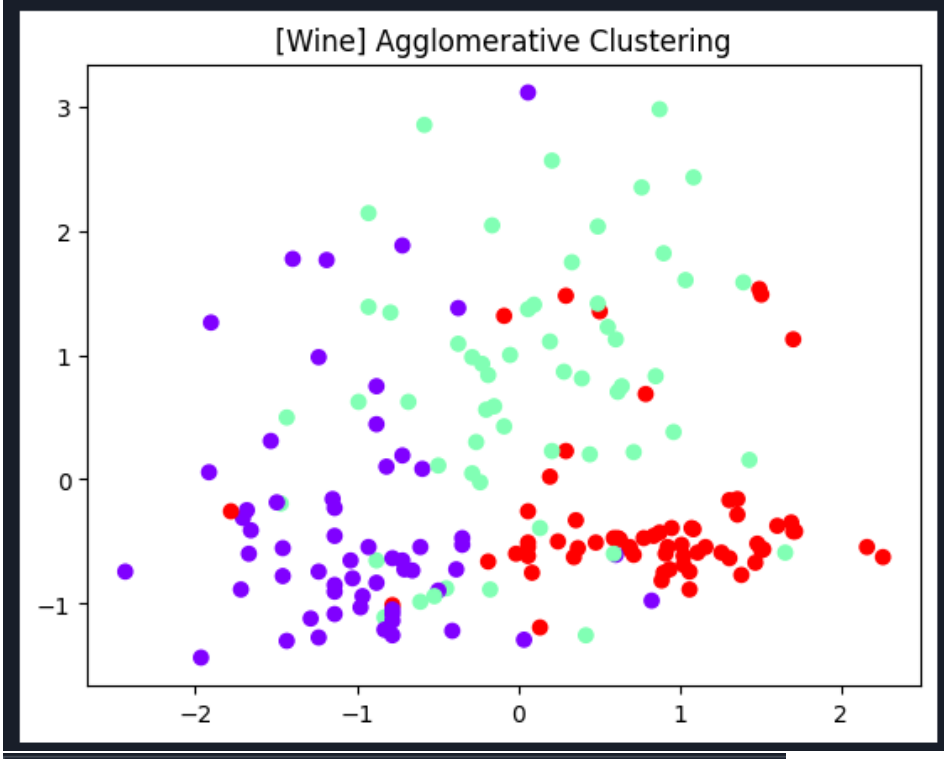



```python
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.cluster import AgglomerativeClustering

linked = linkage(X, method='ward')
plt.figure(figsize=(8, 5))
dendrogram(linked)
plt.title(f"[{dataset_label}] Dendrogram (ward linkage)")
plt.xlabel("Obserwacje")
plt.ylabel("Odległość")
plt.grid(True)
plt.show()

agg = AgglomerativeClustering(n_clusters=3)
labels_agg = agg.fit_predict(X)
sil_agg = silhouette_score(X, labels_agg)

print(f"[{dataset_label}] Agglomerative Silhouette Score: {sil_agg:.2f}")

plt.scatter(X[:, 0], X[:, 1], c=labels_agg, cmap='rainbow')
plt.title(f"[{dataset_label}] Agglomerative Clustering")
plt.show()
```

✓ 0.2s

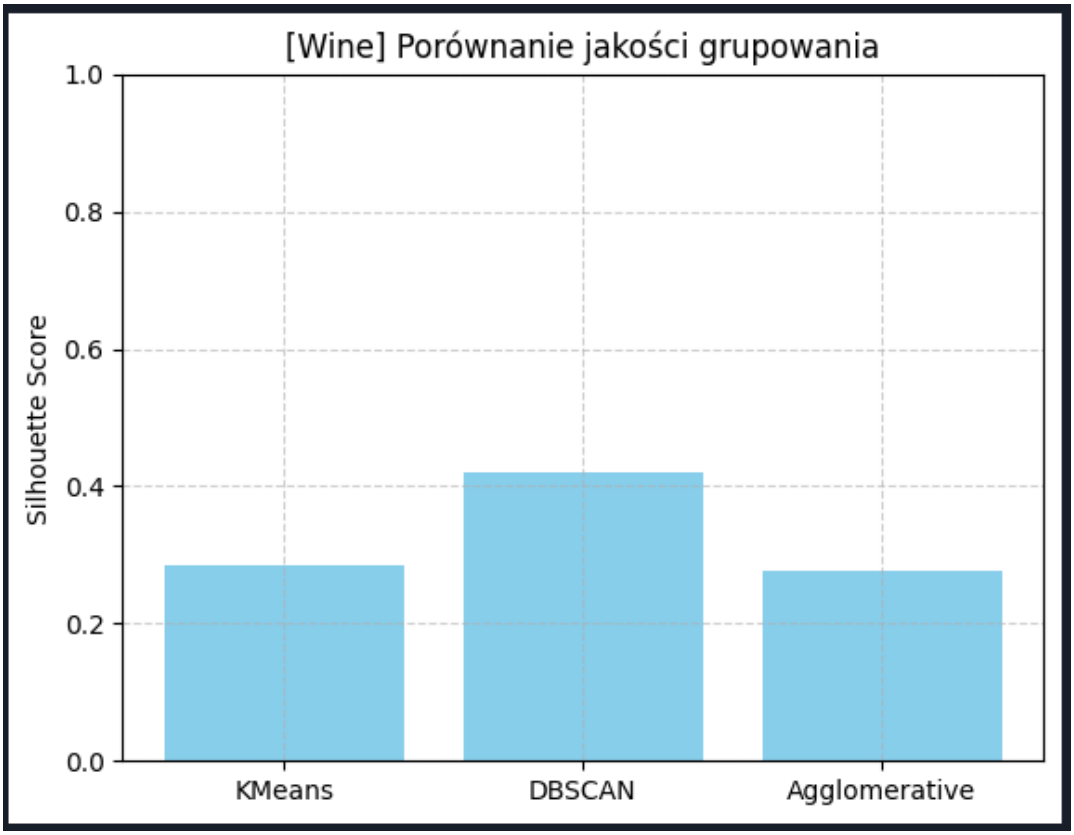



```
[Wine] Agglomerative Silhouette Score: 0.28
```




```
sil_db = 0.42

scores = [sil_km, sil_db, sil_agg]
labels = ['KMeans', 'DBSCAN', 'Agglomerative']

plt.bar(labels, scores, color='skyblue')
plt.ylabel("Silhouette Score")
plt.title(f"[{dataset_label}] Porównanie jakości grupowania")
plt.ylim(0, 1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

✓ 0.0s



3. Wnioski

Wykorzystanie różnych technik grupowania wykazało, że dobór odpowiedniego algorytmu i jego parametrów znacząco wpływa na jakość wyników. Metoda K-średnich okazała się skuteczna przy danych o wyraźnych, kulistych strukturach, natomiast DBSCAN lepiej radził sobie z przypadkami zawierającymi nieregularne skupienia i szum. Z kolei Agglomerative Clustering umożliwił intuicyjną analizę hierarchicznej struktury danych dzięki zastosowaniu dendrogramu.

Ocena efektywności metod z wykorzystaniem wskaźnika Silhouette oraz innych miar, takich jak Calinski-Harabasz i Davies-Bouldin, umożliwiła obiektywne porównanie ich działania. Przeprowadzone testy potwierdziły, że nie istnieje uniwersalne rozwiązanie – każda technika posiada zarówno mocne strony, jak i ograniczenia, a skuteczność zależy od specyfiki danych. Dlatego w praktyce warto wypróbować kilka metod i oprzeć decyzję na analizie wskaźników jakości oraz interpretacji graficznej.